# Matthew Mendoza - Assignment 04

CSC 174 - Section 02

```
    Matthew Mendoza - Assignment 04
    Section 4 - Function
    Function Call
    Function Result
    Section 4 - Procedure
    Procedure: Get_TA
    Procedure: Get_TA Call
    Procedure: Get_TA Result
    Procedure: GetStudentCourse
    Procedure: GetStudentCourse Call
```

Procedure : GetStudentCourse Result

A PDF file include screenshots (in the following order):

- 1. Call the function and the results displayed.
- 2. Call the procedures and the results displayed.

### Section 4 - Function

```
/* Function Course Instructor
* Input parameter : course name
 * Return : Instructor name
*/
DELIMITER $$
CREATE FUNCTION Course Instructor (course name VARCHAR(30))
RETURNS VARCHAR(20)
BEGIN
   DECLARE instructor_name VARCHAR(20);
    SELECT InstructorName INTO instructor_name
   FROM Instructor
    -- so we can use the CourseName input parameter in the
    -- WHERE clause of the outer query to return the InstructorName from the
Instructor table
   WHERE InstructorID = (
        SELECT InstructorID
        FROM Course
        WHERE CourseName = course_name
   );
    RETURN instructor_name;
END $$
DELIMITER;
```

```
SELECT Course_Instructor('Marketing Analytics');
                                                      -- Expects : "Byleth
SELECT Course_Instructor('Intro to Prob Theory');
                                                      -- Expects : "Byleth
SELECT Course_Instructor('Computer Forensics');
                                                      -- Expects : "Byleth
SELECT Course_Instructor('Combinatorics');
                                                      -- Expects : "Hanneman von
SELECT Course_Instructor('Critical Thinking');
                                                      -- Expects : "Hanneman von
Essar"
SELECT Course_Instructor('Leadership and Management'); -- Expects: "Hanneman von
Essar"
SELECT Course_Instructor('Nursing Care Of Adults');
                                                     -- Expects : "Manuela
Casagranda"
SELECT Course_Instructor('Microbiology');
                                                      -- Expects : "Manuela
Casagranda"
SELECT Course_Instructor('Argumentation');
                                                      -- Expects : "Manuela
Casagranda"
SELECT Course_Instructor('Operations Management');
                                                      -- Expects : "Shamir
Nevrand"
SELECT Course_Instructor('Principles Of Marketing');
                                                      -- Expects : "Shamir
Nevrand"
SELECT Course_Instructor('Investments');
                                                      -- Expects : "Shamir
Nevrand"
SELECT Course_Instructor('Sustainability Business'); -- Expects : "Cassandra
SELECT Course_Instructor('Venture Growth Strategies'); -- Expects : "Cassandra
SELECT Course Instructor('Corporate Entrepreneurship'); -- Expects : "Cassandra
Charon"
```

#### **Function Result**

```
mysql> SELECT Course_Instructor('Combinatorics'); -- Expects: "Hanneman von Essar"

| Course_Instructor('Combinatorics') |
| Hanneman von Essar |
| Tow in set (0.00 sec)

mysql> SELECT Course_Instructor('Critical Thinking'); -- Expects: "Hanneman von Essar"

| Course_Instructor('Critical Thinking') |
| Hanneman von Essar |
| Hanneman von Essar |
| Tow in set (0.00 sec)

mysql> SELECT Course_Instructor('Leadership and Management'); -- Expects: "Hanneman von Essar"

| Course_Instructor('Leadership and Management'); -- Expects: "Hanneman von Essar"

| Course_Instructor('Leadership and Management') |
| Hanneman von Essar |
| Hanneman von Essar |
| Tow in set (0.00 sec)
```

### Section 4 - Procedure

# Procedure: Get TA

```
/* Logic for procedure Get_TA
Take Instructor_ID as input parameter
   -- Begin procedure body
   -- Print TA names
   -- Course, Student, TA tables are joined together to get TA names for a given
instructor
   -- InstructorID is a foreign key in Course table and a primary key in
Instructor table
   -- CAST() function converts VARCHAR to INT for comparison with INT column in
Course table
   -- End of procedure body
DELIMITER $$ -- Change delimiter to $$ to allow ; in procedure body
CREATE PROCEDURE Get_TA(IN Instructor_ID VARCHAR(20))
BEGIN
    SELECT S.StudentName
    FROM Course AS C, Student AS S, TA AS T
   WHERE C.InstructorID=CAST(Instructor_ID AS UNSIGNED INTEGER) AND C.TASSN=T.SSN
AND T.SSN=S.SSN;
END $$ -- End of procedure body
DELIMITER ; -- Change delimiter back to ;
```

Procedure: Get TA Call

```
CALL Get_TA("0900000001"); -- Expected : Claude and Edelgard

CALL Get_TA("0900000002"); -- Expected : Dimitri and Edelgard

CALL Get_TA("0900000003"); -- Expected : Claude and Dimitri

CALL Get_TA("0900000004"); -- Expected : Claude and Dimitri and Edelgard

CALL Get_TA("0900000005"); -- Expected : Claude and Dimitri and Edelgard
```

### Procedure: Get TA Result

### Procedure: GetStudentCourse

```
/* Logic for procedure GetStudentCourse
Take Course Number as input parameter
   -- Begin procedure body
   -- Print SSN, Student Name, Email
   -- Course, Student, Enrolled tables are joined together to get student info for
a given course
   -- Course_Number is a foreign key in Enrolled table and a primary key in Course
table
   -- End of procedure body
*/
DELIMITER $$
CREATE PROCEDURE GetStudentCourse(IN Course_Number INTEGER)
BEGIN
 SELECT S.SSN, S.StudentName, S.Address, S.Email
 FROM Course AS C, Enrolled AS E, Student AS S
 WHERE Course Number = C.Course No AND C.Course No = E.Course No AND E.SSN ;
END $$
DELIMITER ;
```

#### Procedure: GetStudentCourse Call

```
CALL GetStudentCourse('3300001'); -- Marketing Analytics -> Expect 8 students
CALL GetStudentCourse('3300002'); -- Intro to Prob Theory -> Expect 9 students
CALL GetStudentCourse('3300003'); -- Computer Forensics -> Expect 7 students
CALL GetStudentCourse('3300004'); -- Combinatorics -> Expect 9 students
CALL GetStudentCourse('3300005'); -- Critical Thinking -> Expect 8 students
CALL GetStudentCourse('3300006'); -- Leadership and Management -> Expect 11
students
CALL GetStudentCourse('3300007'); -- Nursing Care Of Adults -> Expect 9 students
CALL GetStudentCourse('3300008'); -- Microbiology -> Expect 7 students
CALL GetStudentCourse('3300009'); -- Argumentation -> Expect 10 students
CALL GetStudentCourse('3300010'); -- Operations Management -> Expect 11 students
CALL GetStudentCourse('3300011'); -- Principles Of Marketing -> Expect 7 students
CALL GetStudentCourse('3300012'); -- Investments -> Expect 9 students
CALL GetStudentCourse('3300013'); -- Sustainability Business -> Expect 1 Student
CALL GetStudentCourse('3300014'); -- Venture Growth Strategies -> Expect 1 Student
```

CALL GetStudentCourse('3300015'); -- Corporate Entrepreneurship -> Expect 1
Student

## Procedure: GetStudentCourse Result

mysql> CALL	GetStudentCourse('3300001')	); Marketing Analytics -> Expe	ect 8 students
SSN	StudentName	Address	Email
19000006   19000008   19000010   19000011   19000014	Haruhi Fujioka   Ferdinand von Aegir   Dorothea Arnault   Mercedes von Bartels   Annette Fantine Dominic   Raphael Kirsten   Yuri Leclerc   Edelgard von Hresvelg	7098 Prince Ave 123 House Aegir 941 Mittelfrank Opera Company 321 Garreg Mach Monastery 572 House Dominic 8544 Leicester 8982 Rowe Ave 8756 Enbarr	haruhi_fujioka@hostclub.com   ferdinand_von_aegir@aegir.com   dorothea@mittelfrank_opera.com   mercie@garregmach.com   annette@dominic.com   the_beast@leicester.com   Yuri@ashen_wolves.com   flame_emperor@adrestian.com
8 rows in se	t (0.00 sec)		++
Query OK, 0	rows affected (0.00 sec)		

+	·	Intro to Prob Theory -> Expect	
SSN +	Kyoya Ootori Takashi Morinozuka Ferdinand von Aegir	Address      7098 Prince Ave   756 Pennington Road   9245 Pineknoll Ave   123 House Aegir   941 Mittelfrank Opera Company   123 House Fraldarius	Email      haruhi_fujioka@hostclub.com   kyoya_ootori@hostclub.com   takashi_nozuka@hostclub.com   ferdinand_von_aegir@aegir.com   dorothea@mittelfrank_opera.com   felix@fraldarius.com
190000015   200100001   200100002	Yuri Leclerc Claude von Riegan Dimitri Alexandre Blaiddyd	8982 Rowe Ave   4592 Leicester Federation   1462 Garreg Mach Monastery	Yuri@ashen_wolves.com   claude@leicester.com   the_tempest_king@fhirdiad.com
9 rows in set Query OK, 0 r	rows affected (0.00 sec)		

mysql> CALL (	GetStudentCourse('3300003');	Computer Forensics -> Expect 7	7 students
SSN	StudentName	Address	Email
190000002   190000003   190000008   190000010   190000013   200100002	Haruhi Fujioka   Tamaki Suoh   Kyoya Ootori   Dorothea Arnault   Mercedes von Bartels   Marianne von Edmund   Dimitri Alexandre Blaiddyd	7098 Prince Ave 743 Arlington Lane 756 Pennington Road 941 Mittelfrank Opera Company 321 Garreg Mach Monastery 8254 House Edmund 1462 Garreg Mach Monastery	haruhi_fujioka@hostclub.com  tamaki_suoh@hostclub.com  kyoya_ootori@hostclub.com  dorothea@mittelfrank_opera.com  mercie@garregmach.com  Marianne@Edmund.com  the_tempest_king@fhirdiad.com
7 rows in set Query OK, 0 r	rows affected (0.00 sec)		

mysql> CALL (	GetStudentCourse('3300006');	Leadership and Management -	-> Expect 11 students
SSN	StudentName	Address	Email
190000004   190000005   190000007   190000009   190000011   190000012   190000013   200100001	Kyoya Ootori   Mitsukuni Haninozuka   Takashi Morinozuka   Bernadetta von Varley   Felix Hugo Fraldarius   Annette Fantine Dominic   Lysithea von Ordelia   Marianne von Edmund   Claude von Riegan   Dimitri Alexandre Blaiddyd   Edelgard von Hresvelg	756 Pennington Road 142 Old Vernon St 9245 Pineknoll Ave 123 House Varley 123 House Fraldarius 572 House Dominic 9245 House Ordelia 8254 House Edmund 4592 Leicester Federation 1462 Garreg Mach Monastery 8756 Enbarr	kyoya_ootori@hostclub.com   mitsukuni_nozuka@hostclub.com   takashi_nozuka@hostclub.com   bernie@varley.com   felix@fraldarius.com   annette@dominic.com   lys@ordelia.com   Marianne@Edmund.com   claude@leicester.com   the_tempest_king@fhirdiad.com   flame_emperor@adrestian.com
11 rows in s	+et (0.00 sec)	<b>!</b>	<del> </del> +
Query OK, 0	rows affected (0.00 sec)		

mysql> CALL (	GetStudentCourse('330000)	7'); Nursing Care Of Adults ->	Expect 9 students
SSN	StudentName	Address	Email
190000010   190000014   190000015	Mitsukuni Haninozuka   Ferdinand von Aegir   Bernadetta von Varley   Dorothea Arnault   Mercedes von Bartels	7098 Prince Ave 142 Old Vernon St 123 House Aegir 123 House Varley 941 Mittelfrank Opera Company 321 Garreg Mach Monastery 8544 Leicester 8982 Rowe Ave 4592 Leicester Federation	haruhi_fujioka@hostclub.com   mitsukuni_nozuka@hostclub.com   ferdinand_von_aegir@aegir.com   bernie@varley.com   dorothea@mittelfrank_opera.com   mercie@garregmach.com   the_beast@leicester.com   Yuri@ashen_wolves.com
9 rows in se	t (0.00 sec) rows affected (0.00 sec)		•

mysql> CALL ( +	GetStudentCourse('3300008'); - +	Microbiology -> Expect 7 stude +	ents +
190000001   190000004	Haruhi Fujioka   Mitsukuni Haninozuka   Takashi Morinozuka   Dorothea Arnault   Felix Hugo Fraldarius   Raphael Kirsten   Dimitri Alexandre Blaiddyd	7098 Prince Ave   142 Old Vernon St   9245 Pineknoll Ave   941 Mittelfrank Opera Company   123 House Fraldarius   8544 Leicester   1462 Garreg Mach Monastery	haruhi_fujioka@hostclub.com     mitsukuni_nozuka@hostclub.com     takashi_nozuka@hostclub.com     dorothea@mittelfrank_opera.com     felix@fraldarius.com     the_beast@leicester.com     the_tempest_king@fhirdiad.com
7 rows in se	+t (0.00 sec) rows affected (0.00 sec)	<b>!</b>	<b></b>

mysql> CALL (	GetStudentCourse('3300009'); -	Argumentation -> Expect 10	students
SSN	StudentName	Address	Email
190000011   190000012   190000013   190000014   200100002   200100003	Mitsukuni Haninozuka Bernadetta von Varley Mercedes von Bartels Annette Fantine Dominic Lysithea von Ordelia Marianne von Edmund Raphael Kirsten Dimitri Alexandre Blaiddyd Edelgard von Hresvelg	743 Arlington Lane 142 Old Vernon St 123 House Varley 321 Garreg Mach Monastery 572 House Dominic 9245 House Ordelia 8254 House Edmund 8544 Leicester 1462 Garreg Mach Monastery 8756 Enbarr	tamaki_suoh@hostclub.com  mitsukuni_nozuka@hostclub.com  bernie@varley.com  mercie@garregmach.com  annette@dominic.com  lys@ordelia.com  Marianne@Edmund.com  the_beast@leicester.com  flame_emperor@adrestian.com
10 rows in se	et (0.00 sec) rows affected (0.00 sec)		

```
mysql> CALL GetStudentCourse('3300010'); -- Operations Management -> Expect 11 students
  SSN
                StudentName
                                                   Address
                                                                                  | Email
  19000001 | Haruhi Fujioka
190000002 | Tamaki Suoh
                                                                                  | haruhi fujioka@hostclub.com
                                                   | 7098 Prince Ave
  190000002 | Tamaki Suoh | 743 Arlington Lane | tamaki_suoh@hostclub.
190000004 | Mitsukuni Haninozuka | 142 Old Vernon St | mitsukuni_nozuka@hostclub.
190000005 | Takashi Morinozuka | 9245 Pineknoll Ave | takashi_nozuka@hostcl
190000007 | Bernadetta von Varley | 123 House Varley | bernie@varley.com
190000009 | Felix Hugo Fraldarius | 123 House Fraldarius | felix@fraldarius.com
                                                                                  | tamaki_suoh@hostclub.com
                                                                                  mitsukuni nozuka@hostclub.com
                                                                                  takashi_nozuka@hostclub.com
  190000011 | Annette Fantine Dominic | 572 House Dominic | annette@dominic.com
  190000012 | Lysithea von Ordelia | 9245 House Ordelia
                                                                                  | lys@ordelia.com
                                                 | 8544 Leicester
| 8982 Rowe Ave
  190000014 | Raphael Kirsten
                                                                                    the beast@leicester.com
  190000015 | Yuri Leclerc
                                                                                  | Yuri@ashen wolves.com
  200100003 | Edelgard von Hresvelg | 8756 Enbarr
                                                                                  | flame_emperor@adrestian.com
11 rows in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
```

190000006   Ferdinand von Aegir   123 House Aegir   ferdinand_von_aegir@aegir.c 190000007   Bernadetta von Varley   123 House Varley   bernie@varley.com 190000010   Mercedes von Bartels   321 Garreg Mach Monastery   mercie@garregmach.com 190000011   Annette Fantine Dominic   572 House Dominic   annette@dominic.com 190000012   Lysithea von Ordelia   9245 House Ordelia   lys@ordelia.com	SSN	StudentName	Address	Email
190000015   Yuri Leclerc   8982 Rowe Ave   Yuri@ashen_wolves.com	190000007 190000010 190000011 190000012 190000013	Ferdinand von Aegir Bernadetta von Varley Mercedes von Bartels Annette Fantine Dominic Lysithea von Ordelia Marianne von Edmund	123 House Aegir 123 House Varley 321 Garreg Mach Monastery 572 House Dominic 9245 House Ordelia 8254 House Edmund	ferdinand_von_aegir@aegir.com   bernie@varley.com   mercie@garregmach.com   annette@dominic.com   lys@ordelia.com   Marianne@Edmund.com

SSN	StudentName	Address	Email
190000008 190000009 190000012 190000014	Takashi Morinozuka Dorothea Arnault Felix Hugo Fraldarius Lysithea von Ordelia	756 Pennington Road 142 Old Vernon St 9245 Pineknoll Ave 941 Mittelfrank Opera Company 123 House Fraldarius 9245 House Ordelia 8544 Leicester 8982 Rowe Ave	kyoya_ootori@hostclub.com   mitsukuni_nozuka@hostclub.com   takashi_nozuka@hostclub.com   dorothea@mittelfrank_opera.com   felix@fraldarius.com   lys@ordelia.com   the_beast@leicester.com   Yuri@ashen_wolves.com
rows in set	+ t (0.00 sec)		<del> </del>